UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/711,733	09/30/2004	Lee George Laborczfalvi	2006579-0143	5732

69665 7590 11/12/2009

CHOATE, HALL & STEWART / CITRIX SYSTEMS, INC.
TWO INTERNATIONAL PLACE
BOSTON, MA 02110

EXAMINER	
ABDUL-ALI, OMAR R	

ART UNIT	PAPER NUMBER
2173	

MAIL DATE	DELIVERY MODE
11/12/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/711,733	LABORCZFALVI ET AL.	
	Examiner	Art Unit	
	OMAR ABDUL-ALI	2173	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 July 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-29* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-29* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This action is in response to the response filed 7/01/2009. Amended Claims 1-29 are pending and have been considered below.

1. The prior art rejections have been withdrawn as necessitated by applicant's amendments.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. Claims 14-19 remain rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 14-19 are drawn to a computer program per se. A computer program is not a series of steps or acts and this is not a process. A computer program is not a physical article or object and as such is not a machine or manufacture. A computer program is not a combination of substances and therefore not a compilation of matter. The apparatus, as claimed, does not include any statutory features (processor, computer display, for example), and only contains software components such as a hooking mechanism, window name virtualization engine, and operating system interface. Thus, a computer program by itself does not fall within any of the four categories of invention. Therefore, Claims 14-19 are not statutory.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-17, and 19-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Parker et al. (US 5,781,720) in view of Demsey et al. (US 7,203,941).

Claim 1: Parker discloses a method for virtualizing access to windows, the method comprising receiving a request related to a window from a process the request including a virtual window name (logical name) (column 13, 1-25). However, Parker does not explicitly disclose receiving the request within the context of a user isolation scope, wherein the user isolation scope is provided by an isolation environment comprising a user isolation layer and an application isolation layer. Demsey discloses a similar method that further discloses receiving a request in the context of a user isolation scope (column 5, lines 40-50; The managed code portion includes virtual machine (VM) 104 and App (k) 102 in user code), the isolation environment (interface 110 between a managed code portion and a native code portion) including a user isolation layer (user code) and application isolation layer (managed code portion). The system parameter calls generated by applications and user input initiate calls for native resources across the interface between the native code portion and the managed code portion (column 5, lines 51-67). Therefore, it would have been obvious to one having ordinary skill in the

art at the time the invention was made to include a user isolation scope provided by an isolation environment comprising a user isolation layer and application isolation layer in Parker, for the purpose of enhancing a user's experience of programs calling for native resources through greater interoperability between computer environments.

Parker modified by Demsey discloses determining a literal name (GUI specific name) for the window using a scope-specific identifier associated with at least one of a particular user isolation scope and an application isolation scope. Parker discloses a test script specifies a request against a logically named LSE (window), and a test executive resolves the LSE's logical name contained in the script command into a GUI specific name as a parameter (column 13, lines 1-25).

Parker does not explicitly disclose issuing to the operating system a request including the determined literal name. However, Demsey further discloses using system calls made when executing code in the virtual machine environment, where each caller makes a call through the operating system (column 6, lines 57-61). It would have been obvious to one having ordinary skill in the art at the time the invention was made to issue to the operating system a request including a determined literal name in Parker. One would have been motivated to issue the request to the operating system in order to retrieve user interface elements that are managed by the operating system.

Parker modified by Demsey discloses associating a window handle (tag) with the determined virtual window name (column 19, lines 40-50).

Claim 2: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving a request further comprises intercepting a request relating to a window from a process executing in the context of a user isolation scope, the request including a virtual window name (Figure 3, Application Executing in Virtual Machine Makes A Request in Managed Code for Native Resource Access).

Claim 3: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving a request further comprises receiving a request to find a window from a process executing in the context of a user isolation scope, the request including a virtual window name (Figure 3, Application Executing in Virtual Machine Makes A Request in Managed Code for Native Resource Access).

Claim 4: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses receiving a request further comprises receiving a request to create a window from a process executing in the context of a user account, the request including a virtual window name (Figure 3, Application Executing in Virtual Machine Makes A Request in Managed Code for Native Resource Access).

Claim 5: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses determining a rule associated with the

virtual window name included in the request and determining a literal name for the window responsive to the determined rule (column 13, lines 1-25).

Claim 6: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses determining a literal name further comprises determining a literal window name using a scope-specific identifier associated with an application isolation scope with which the process making the request is associated (column 133, lines 1-25).

Claim 7: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses associating a window handle further comprises storing the virtual window name in a mapping table associated with a window handle (column 23, lines 11-23).

Claim 8: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving from the operating system a response to the issued request (column 6, lines 57-67). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to receive a response from the operating system in Parker. One would have been motivated to receive a response from an operating system in order to retrieve user interface elements that are managed by the operating system.

Claim 9: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further disclose replacing the literal window name in the response with a virtual window name (column 26, lines 1-15).

Claim 10: Parker discloses a method for virtualizing access to windows, comprising receiving a request to identify one of a virtual window name and a virtual window class identifier, the request received from a process executing within the context of a user account and including a window handle (tag). Parker discloses a test script specifies a request against a logically named LSE (window), and a test executive resolves the LSE's logical name contained in the script command into a GUI specific name as a parameter (column 13, lines 1-25). However, Parker does not explicitly disclose receiving the request within the context of a user isolation scope, wherein the user isolation scope is provided by an isolation environment comprising a user isolation layer and an application isolation layer. Demsey discloses a similar method that further discloses receiving a request in the context of a user isolation scope (column 5, lines 40-50; The managed code portion includes virtual machine (VM) 104 and App (k) 102 in user code), the isolation environment (interface 110 between a managed code portion and a native code portion) including a user isolation layer (user code) and application isolation layer (managed code portion). The system parameter calls generated by applications and user input initiate calls for native resources across the interface between the native code portion and the managed code portion (column 5, lines 51-67). Therefore, it would have been obvious to one having ordinary skill in the art at the time

the invention was made to include a user isolation scope provided by an isolation environment comprising a user isolation layer and application isolation layer in Parker, for the purpose of enhancing a user's experience of programs calling for native resources through greater interoperability between computer environments.

Parker modified by Demsey discloses determining that the window handle (tag) is associated with the requested one of the virtual window name and the virtual window class identifier (column 19, lines 40-50).

Parker modified by Demsey discloses returning to the requesting process the determined window information (column 13, lines 1-25).

Claim 11: Parker and Demsey disclose a method for virtualizing access to windows as in claim 10 above, and Parker further discloses determining that the window handle is associated with the requested window name further comprises determining whether an association between the window handle and the requested one of the virtual window name and the virtual window class identifier exists (column 23, lines 10-22).

Claim 12: Parker and Demsey disclose a method for virtualizing access to windows as in claim 11 above, and Parker further discloses determining the window handle associated with the requested one of the virtual name and the virtual window class identifier from a mapping table, responsive to determining that an association exists in the mapping table (column 23, lines 10-22)

Claim 13: Parker and Demsey disclose a method of virtualizing access to windows as in claim 11 above, and Demsey further discloses returning to the requesting process a response received from an operating system responsive to determining no association exists in the mapping table (column 7, lines 52-62). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to receive a response from the operating system in Parker. One would have been motivated to receive a response from an operating system in order to retrieve user interface elements that are managed by the operating system.

Claim 14: Parker discloses a method for virtualizing access to windows, the method comprising a hooking mechanism receiving a request related to a window from a process, the request including one of a virtual window name (logical name) and a virtual window class identifier (column 13, 1-25). However, Parker does not explicitly disclose receiving the request within the context of a user isolation scope, wherein the user isolation scope is provided by an isolation environment comprising a user isolation layer and an application isolation layer. Demsey discloses a similar method that further discloses receiving a request in the context of a user isolation scope (column 5, lines 40-50; The managed code portion includes virtual machine (VM) 104 and App (k) 102 in user code), the isolation environment (interface 110 between a managed code portion and a native code portion) including a user isolation layer (user code) and application isolation layer (managed code portion). The system parameter calls generated by applications and user input initiate calls for native resources across the interface

between the native code portion and the managed code portion (column 5, lines 51-67). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to include a user isolation scope provided by an isolation environment comprising a user isolation layer and application isolation layer in Parker, for the purpose of enhancing a user's experience of programs calling for native resources through greater interoperability between computer environments.

b. a window name virtualizaton engine forming one of a literal name for the window and a literal class identifier using one of the virtual window name and the virtual window class identifier received in the request and a scope specific identifier associated with a particular isolation scope. Parker discloses a test script specifies a request against a logically named LSE (window), and a test executive resolves the LSE's logical name contained in the script command into a GUI specific name as a parameter (column 13, lines 1-25).

Parker does not explicitly disclose an operating system interface issuing a request relating to a window, the request including the one of the formed literal name and the formed literal window class identifier for the window. However, Demsey further discloses using system calls made when executing code in the virtual machine environment, where each caller makes a call through the operating system (column 6, lines 57-61). It would have been obvious to one having ordinary skill in the art at the time the invention was made to issue to the operating system a request including a determined literal name in Parker. One would have been motivated to issue the request

to the operating system in order to retrieve user interface elements that are managed by the operating system.

Claim 15: Parker and Demsey disclose a method for virtualizing access to windows, and Parker further discloses the hooking mechanism intercepts a request selected from a group consisting of finding a window, creating a window, enumerating a window, destroying a window, setting a window name, retrieving a window name, retrieving a window class identifier associated with the window, registering a window class, retrieving information about a window class and unregistering a window class (column 13, lines 1-25).

Claim 16: Parker and Demsey disclose a method for virtualizing access to windows, and Parker further discloses a mapping table storing an association between a window handle and one of the virtual window name and the virtual window class identifier (column 23, lines 11-21).

Claim 17: Parker and Demsey disclose a method for virtualizing access to windows, and Parker further discloses the mapping table is associated with the process (column 23, lines 11-21).

Claim 19: Parker and Demsey disclose a method for virtualizing access to windows, and Parker further discloses a rules engine comprising a rule determining how the

window virtualization engine forms the one of the literal name for the window and the literal class identifier for the window (column 13, lines 1-25).

Claim 20: Parker discloses a method for virtualizing access to windows, the method comprising intercepting a request, from a requester, to paint a title bar for a window, the title bar including the window name, the request including a window handle (column 10, lines 29-42) Parker discloses a test script reads a windows name through a title bar. However, Parker does not explicitly disclose a requestor executing within the context of an isolation scope, the isolation scope provided by an isolation environment comprising a user isolation layer and an application isolation layer. Demsey discloses a similar method that further discloses receiving a request in the context of a user isolation scope (column 5, lines 40-50; The managed code portion includes virtual machine (VM) 104 and App (k) 102 in user code), the isolation environment (interface 110 between a managed code portion and a native code portion) including a user isolation layer (user code) and application isolation layer (managed code portion). The system parameter calls generated by applications and user input initiate calls for native resources across the interface between the native code portion and the managed code portion (column 5, lines 51-67). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to include a user isolation scope provided by an isolation environment comprising a user isolation layer and application isolation layer in Parker, for the purpose of enhancing a user's experience of programs calling for native resources through greater interoperability between computer environments.

b. determining that the window handle (tag) is associated with the virtual window name (column 20, lines 29-39);

c. painting the title bar of the window using the virtual window name (column 20, lines 29-39);

d. indicating to the requestor that the title bar has been painted (column 22, lines 58-68).

Claim 21: Parker discloses a method for virtualizing access to windows, the method comprising receiving a request, relating to a window class (superclass), from a process, the request including a virtual window class identifier (column 13, lines 1-25). However, Parker does not explicitly disclose a requestor executing within the context of an isolation scope, the isolation scope provided by an isolation environment comprising a user isolation layer and an application isolation layer. Demsey discloses a similar method that further discloses receiving a request in the context of a user isolation scope (column 5, lines 40-50; The managed code portion includes virtual machine (VM) 104 and App (k) 102 in user code), the isolation environment (interface 110 between a managed code portion and a native code portion) including a user isolation layer (user code) and application isolation layer (managed code portion). The system parameter calls generated by applications and user input initiate calls for native resources across the interface between the native code portion and the managed code portion (column 5, lines 51-67). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to include a user isolation scope provided by an

isolation environment comprising a user isolation layer and application isolation layer in Parker, for the purpose of enhancing a user's experience of programs calling for native resources through greater interoperability between computer environments.

b. determining a literal window class identifier using a scope specific identifier associated with a particular isolation scope Parker discloses a test script specifies a request against a logically named LSE (window), and a test executive resolves the LSE's logical name contained in the script command into a GUI specific name as a parameter (column 13, lines 1-25).

Parker does not explicitly disclose issuing to an operating system a request including the determined literal window class identifier. However, Demsey further discloses using system calls made when executing code in the virtual machine environment, where each caller makes a call through the operating system (column 6, lines 57-61). It would have been obvious to one having ordinary skill in the art at the time the invention was made to issue to the operating system a request including a determined literal name in Parker. One would have been motivated to issue the request to the operating system in order to retrieve user interface elements that are managed by the operating system.

Parker modified by Demsey discloses associating a window handle (tag) with the determined literal window class identifier (column 19, lines 40-50).

Claim 22: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving a request further comprises

intercepting a request relating to a window class from a process executing in the context of a user isolation scope, the request including a virtual window class identifier (Figure 3, Application Executing in Virtual Machine Makes A Request in Managed Code for Native Resource Access).

Claim 23: Parker and Oppermann disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving a request further comprises receiving a request to find a window from a process executing in the context of a user isolation scope, the request including a virtual window class identifier (Figure 3, Application Executing in Virtual Machine Makes A Request in Managed Code for Native Resource Access).

Claim 24: Parker and Oppermann disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving a request further comprises receiving a request to create a window from a process executing in the context of a user isolation scope, the request including a virtual window class identifier (Figure 3, Application Executing in Virtual Machine Makes A Request in Managed Code for Native Resource Access).

Claim 25: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses determining a rule associated with the

virtual window class identifier included in the request and determining a literal name for the window responsive to the determined rule (column 13, lines 1-25).

Claim 26: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses determining a literal name further comprises determining a literal window class name using a scope-specific identifier associated with an application isolation scope with which the process making the request is associated (column 133, lines 1-25).

Claim 27: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses associating a window handle further comprises storing the virtual window class identifier in a mapping table associated with a window handle (column 23, lines 11-23).

Claim 28: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Demsey further discloses receiving from the operating system a response to the issued request (column 6, lines 57-67). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to receive a response from the operating system in Parker. One would have been motivated to receive a response from an operating system in order to retrieve user interface elements that are managed by the operating system.

Claim 29: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, and Parker further discloses replacing the literal window name in the response with a virtual window name (column 26, lines 1-15)).

6. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over Parker et al. (US 5,781,720) in view of Demsey et al. (US 7,203,941) and further in view of Craycroft (US 5,856,826).

Claim 18: Parker and Demsey disclose a method of virtualizing access to windows as in claim 1 above, but neither reference explicitly discloses a second mapping table associated with a second process. Demsey discloses a similar system that further discloses maintaining window data in multiple mapping tables (column 7, lines 5-15). It would have been obvious to one having ordinary skill in the art at the time the invention was made to include a second mapping table associated with a second processing because the use of multiple mapping tables is a known technique in the computer arts. One would have been motivated to include a second mapping table in order to increase efficiency.

Response to Arguments

7. Applicant's arguments with respect to claims 1-29 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

8. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to OMAR ABDUL-ALI whose telephone number is (571)270-1694. The examiner can normally be reached on Mon-Fri(Alternate Fridays Off) 9:30 - 7:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kieu Vu can be reached on 571-272-4057. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

OAA
11/06/2009

/Kieu Vu/
Supervisory Patent Examiner, Art Unit 2173